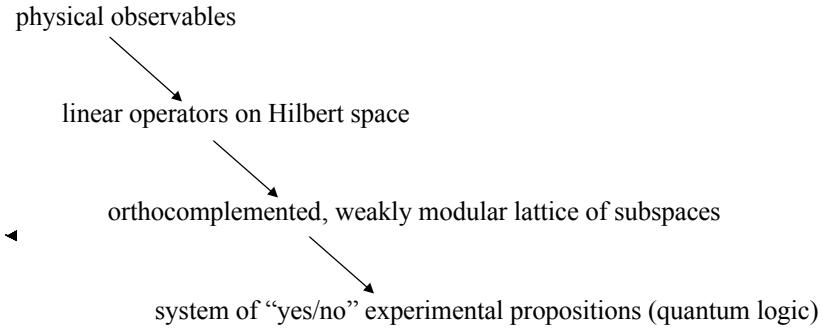
Using Automated Theorem-Provers to Aid the Design of Efficient Compilers for Quantum Computing

Jack K. Horner

Science Applications International Corporation
High Performance Computing Environments (CCN-8)
MS T080
Los Alamos National Laboratory
Los Alamos, NM 87545
(505) 665-2400
jkh@lanl.gov

The doctrine of quantum logic: from observables to propositions



Problem statement

- Conventional programming languages (e.g., C++ or Fortran) are based on Boolean logic (BL; e.g., first-order predicate calculus)
- Quantum logic (QL) is profoundly non-Boolean (e.g., the Boolean distributive law does not hold in QL)
- Quantum computers are at their foundation characterized by QL
- In quantum computing, we seek to optimize the performance of QL-oriented machine code that captures BL-based programs (a compiler optimization problem)
- Because QL and BL are ***not*** isomorphic, optimizing the operations of a BL-based program is not the same as optimizing the representation of that program in QL

Research approach

- Use automated quantum-logic theorem provers to help identify BL-to-QL mapping-optimization opportunities (e.g., to aid the search for shortest derivations of QL theorems)
- Current status: developed *bvn*, an automated theorem-prover for Birkhoff-von Neumann quantum logic (BVNQL)
  - portable, implemented in lex, yacc, and C
  - backtracking-based
  - automatically generates all BVNQL derivations of length less than *l* (user-specified), in time less than *t_max* (user-specified), of a given BVNQL proposition A from a given BVNQL proposition B
- Open issues
  - QLs are not known to be complete
  - most backtracking algorithms viciously sacrifice performance to generality; none ultimately escape this trade